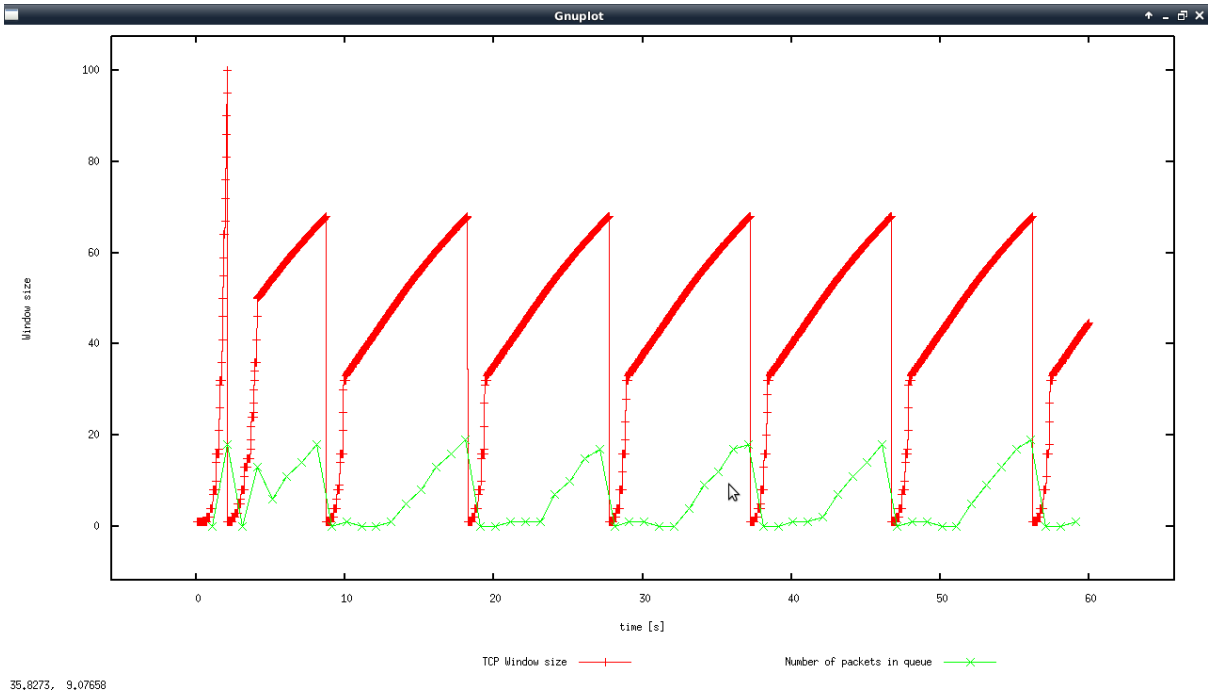
Exercise 1: Understanding TCP Congestion Control using ns-2

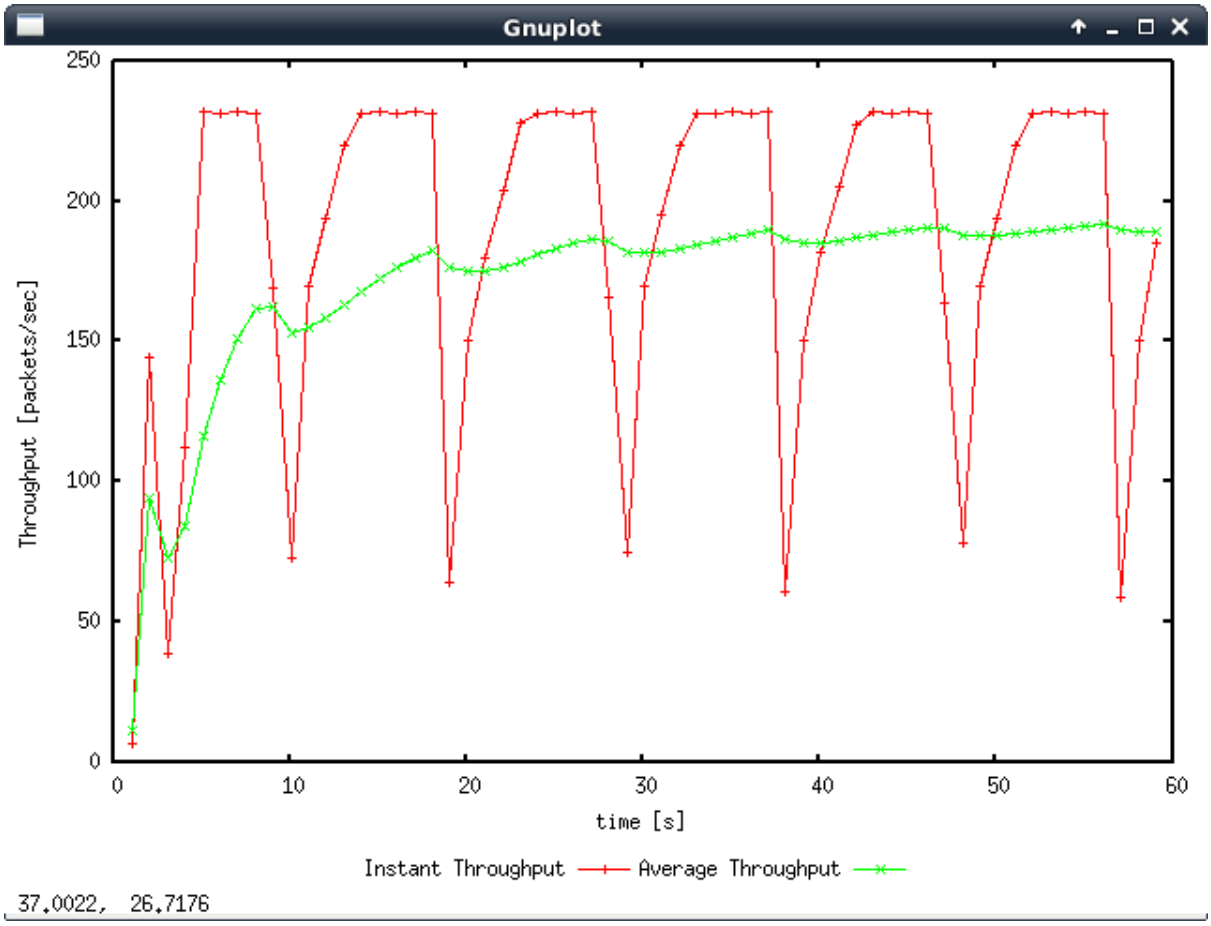
Question 1: What is the maximum size of the congestion window that the TCP flow reaches in this case? What does the TCP flow do when the congestion window reaches this value? Why? What happens next? Include the graph in your submission report.

The maximum congestion window size reached is 100. When the congestion window reaches this value, it experiences a congestion event. This occurs as congestion window size has become too big and due to too many incoming packets, the buffer overflows and packets are lost. This is shown from the green line in the graph below when the congestion window reaches its peak, the buffer (green line) also reaches its peak. As the TCP version being run in the program is Tahoe, once this event occurs, the congestion window returns to the size of 1 (shown in the drop of the red line's value).



Question 2: From the simulation script we used, we know that the payload of the packet is 500 Bytes. Keep in mind that the size of the IP and TCP headers is 20 Bytes, each. Neglect any other headers. What is the average throughput of TCP in this case? (both in number of packets per second and bps)

The average throughput of TCP in this case is 189 packets per second. Including the headers, each packet is (500 + 20) = 520 bytes. Therefore, the throughput in bps is, 189 * 520 = 98280 bytes/second



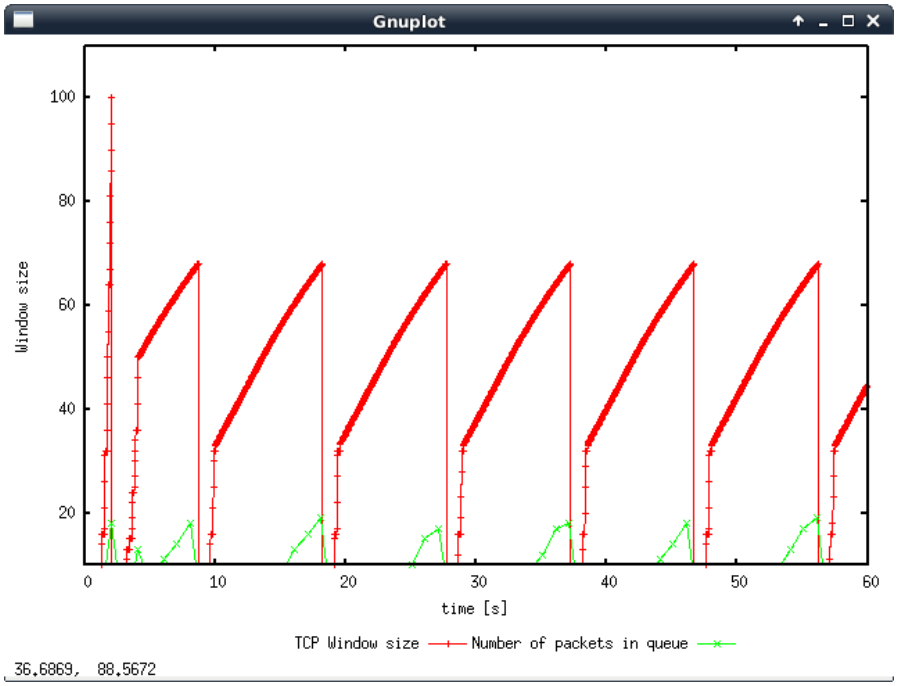
Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT (i.e. 100ms). How does TCP respond to the variation of this parameter? Find the value of the maximum congestion window at which TCP stops oscillating (i.e., does not move up and down again) to reach a stable behaviour. What is the average throughput (in packets and bps) at this point? How does the actual average throughput compare to the link capacity (1Mbps)?

As this parameter decreases, TCP responds by having a more stable congestion window, as its value does not value oscillate as much, and when it does, it usually reaches close to the maximum congestion window value that was inputted. In addition, with lower congestion window values, the number of packets in queue are also reduced. With higher congestion window values, the slow start only reaches a maximum of 100 window size before experiencing congestion events, despite being set to higher window sizes. As the maximum stable value of the congestion window is 66, these higher congestion window values tend to reach this window size through additive increase, before experiencing congestion events.

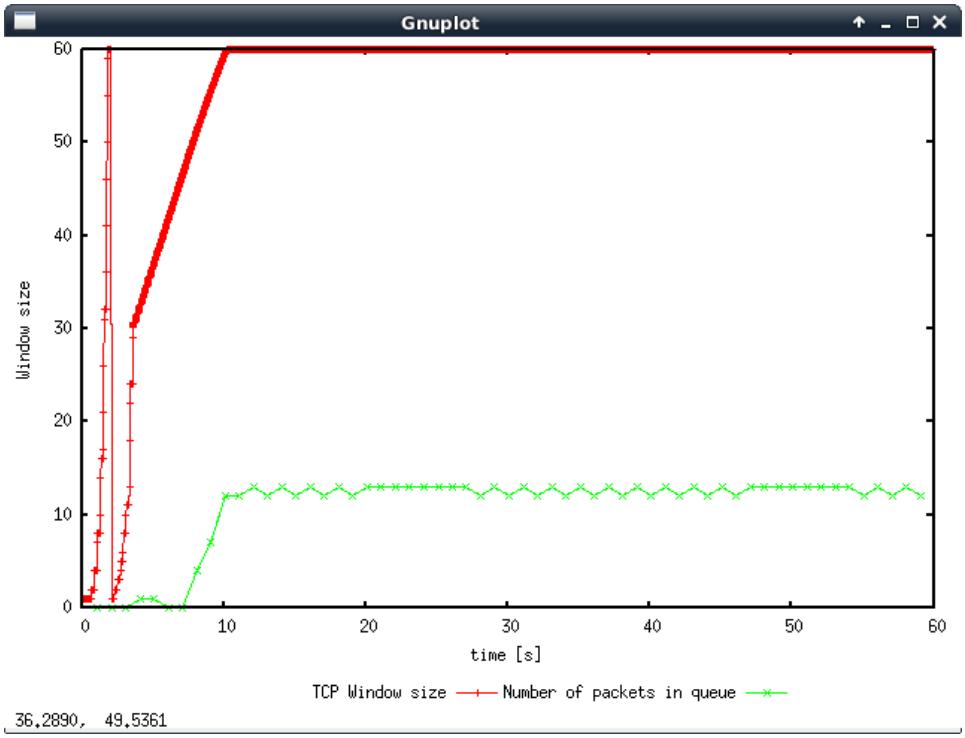
The maximum value of the congestion window is 66 when TCP stops oscillating. The average throughput is 220 packets/ second or 220 * 520 = 114400 bytes /second.

$114400 / 10^6 = 0.1144$ Mb as such the actual throughput only uses ~10% of the link capacity.

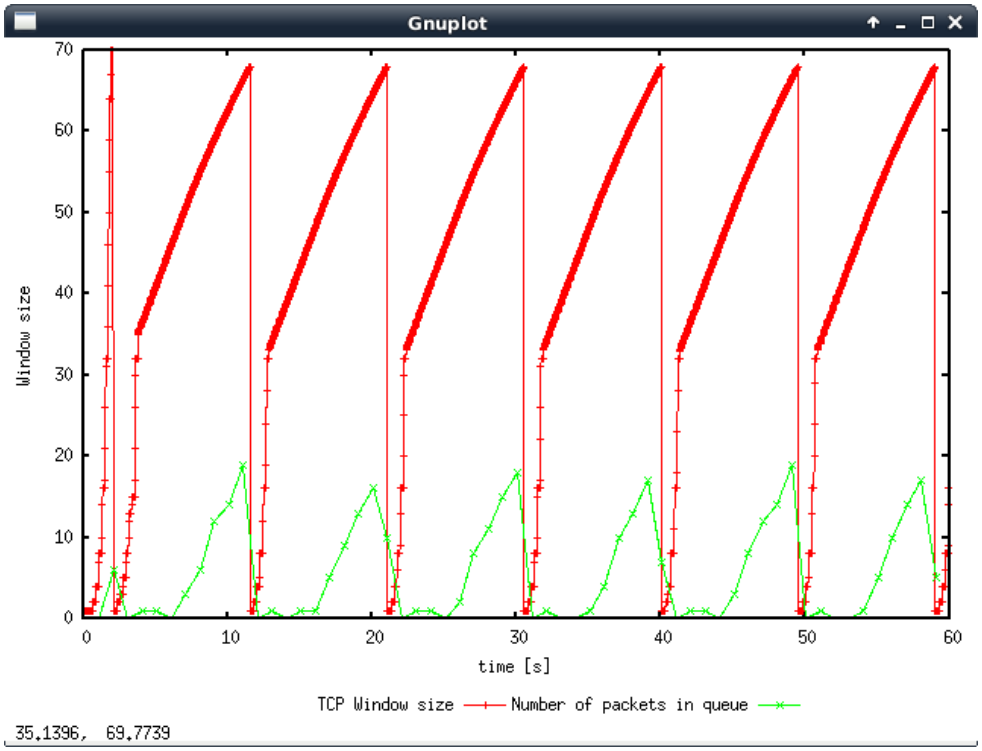
Congestion Window 180



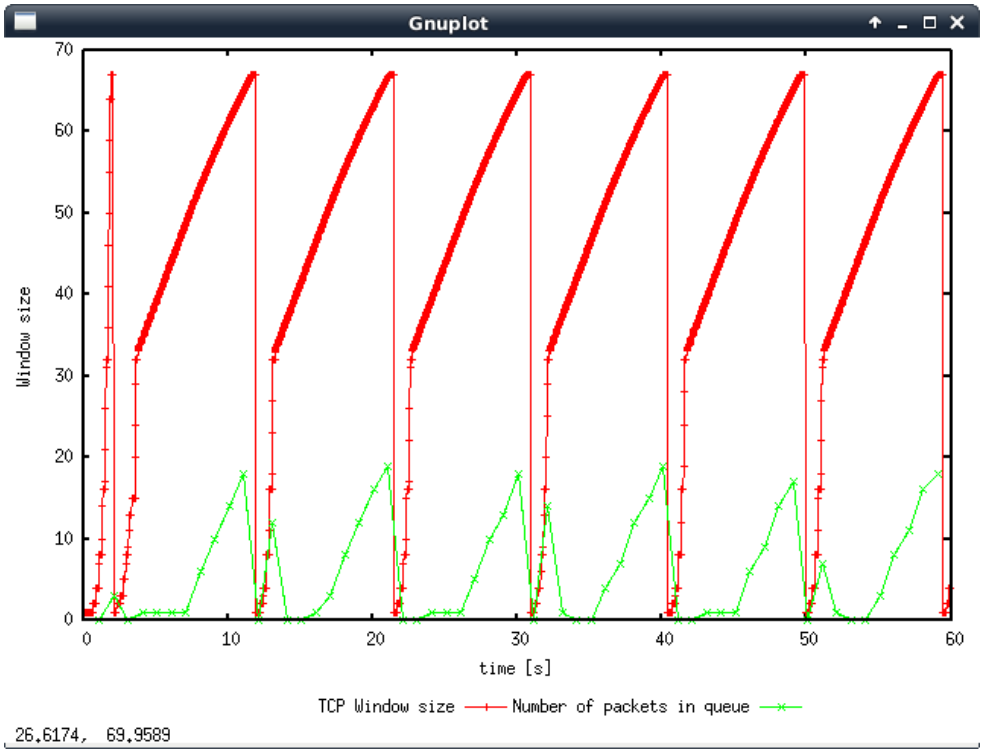
Congestion Window 60



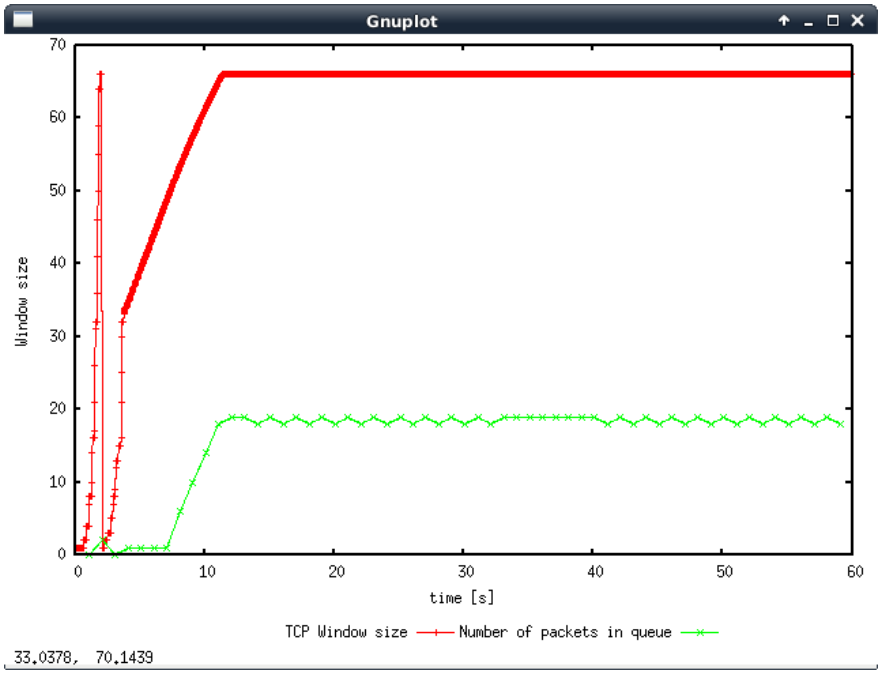
Congestion Window 70



Congestion Window 67



Congestion Window 66

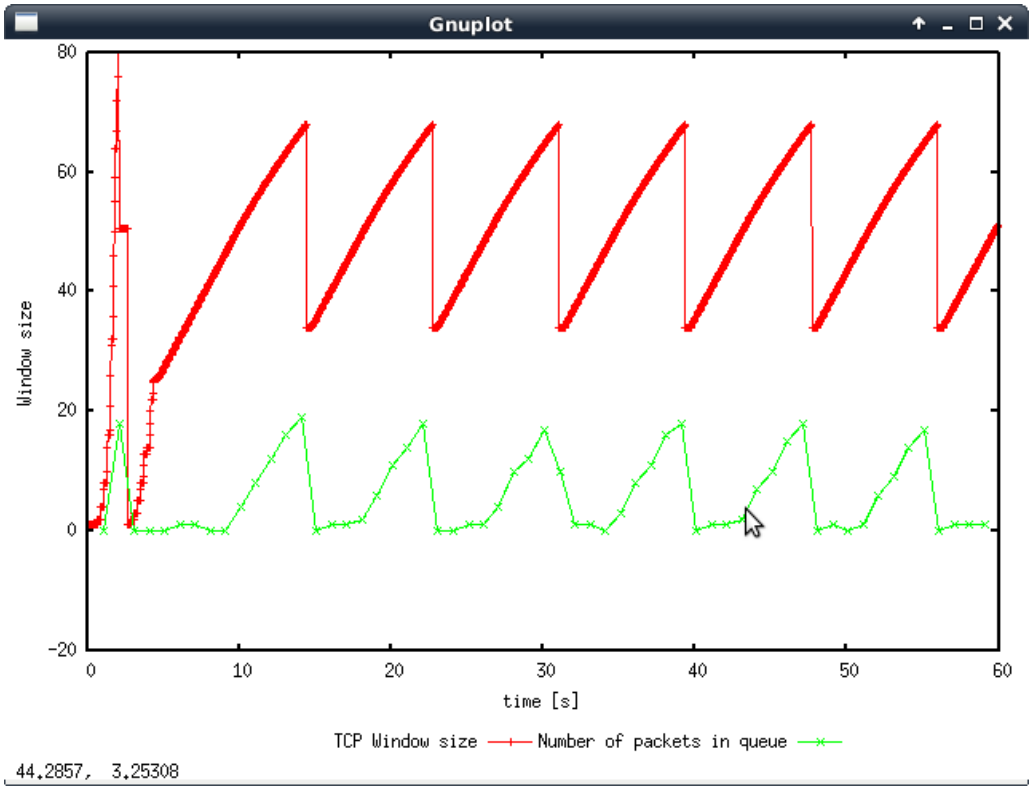


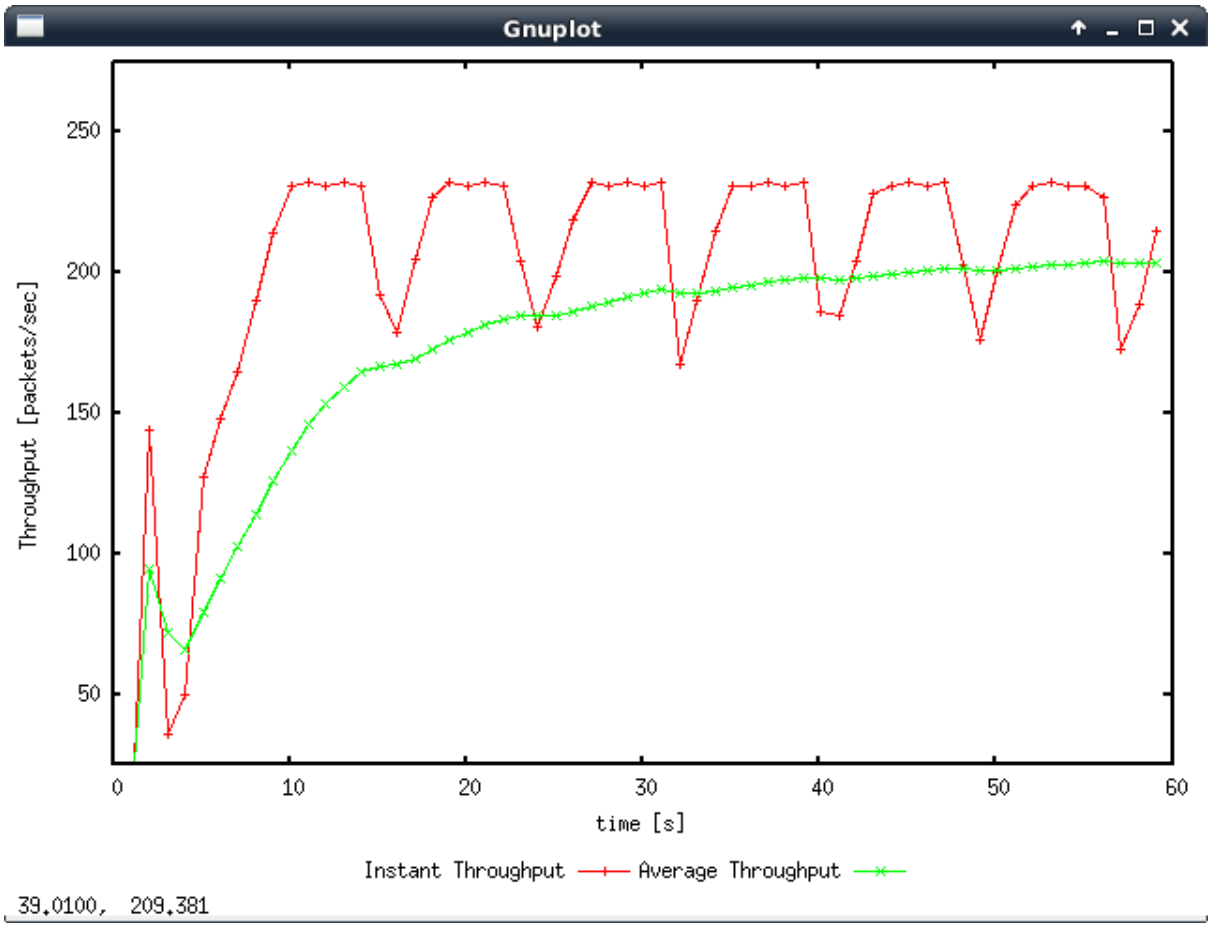
Question 4: Repeat the steps outlined in Question 1 and 2 (NOT Question 3) but for TCP Reno. Compare the graphs for the two implementations and explain the differences. (Hint: compare the number of times the congestion window goes back to zero in each case). How does the average throughput differ in both implementations?

The maximum congestion window for TCP Reno was also 100. Once it reaches this value it experiences a three duplicate ACK event and reduces its congestion window by half, however following this reduction it undergoes another congestion event (Timeout), subsequently reducing its congestion window to zero. This is shown in the graph as the graph pauses at the 50 window size following the first peak before returning to zero.

The average throughput of TCP Reno is 203 packets per second. Including the headers, each packet is (500 + 20) = 520 bytes. Therefore, the throughput in bps is, 203 * 520 = 105560 bytes/second

One of the differences between the two graphs is that for TCP Reno, whenever a congestion event occurs, the congestion window is halved rather than going back to zero as in the case of Tahoe. In addition to this, TCP Reno undergoes additive increase once its congestion window has been reduced. However, for the TCP Tahoe graph, following its congestion window drop, it goes through a slow start before reaching the slow start threshold and continuing to increase its congestion window through additive increase. This creates a difference between the slopes of the two graphs following the first peak.


Gnuplot
Window size
time [s]
TCP Window size
Number of packets in queue
44.2857, 3.25308


Gnuplot
Throughput [packets/sec]
time [s]
Instant Throughput
Average Throughput
39.0100, 209.381

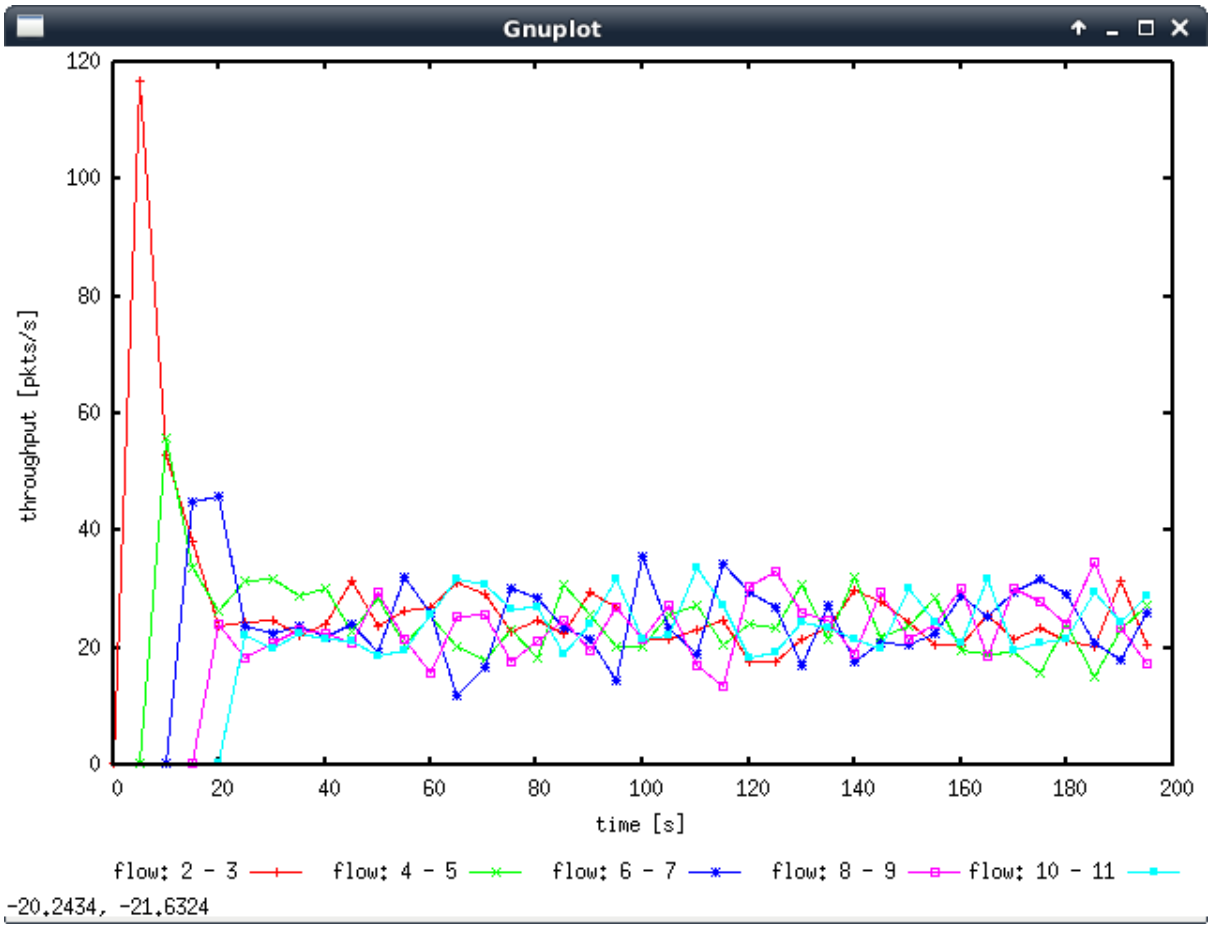
Exercise 2: Flow Fairness with TCP

Question 1: Does each flow get an equal share of the capacity of the common link (i.e., is TCP fair)? Explain which observations lead you to this conclusion.

The TCP flow is relatively fair as when each new flow is added, the throughput of pre-existing flows is reduced. In addition to this, once all flows have entered the link, the flows oscillate around similar values with no particular flow using more/less of the link (as shown in the graph below).

Question 2. What happens to the throughput of the pre-existing TCP flows when a new flow is created? Explain the mechanisms of TCP which contribute to this behaviour. Argue about whether you consider this behaviour to be fair or unfair.

The throughput of pre-existing TCP flows decreases when a new flow is created. TCP contributes to this behaviour as when new flows are created, their congestion window rapidly increases due to TCP's slow start. As the congestion window increases, this new flow uses more and more of the links' bandwidth, preventing previous TCP flows from operating at their previous output. As such pre-existing TCP flows adjust their congestion window size (through AIMD) in order to prevent buffer overflows and packet loss at the bottleneck link. This behaviour is fair as all flows have an equal share in use of the network link.



Exercise 3: TCP competing with UDP

Question 1: How do you expect the TCP flow and the UDP flow to behave if the capacity of the link is 5 Mbps?

I expect that the UDP will take up most of the link bandwidth due to its lack of congestion control. As the TCP flow operates with congestion control, its throughput will be limited by the UDP flow's monopolisation of the link bandwidth.

Question 2: Why does one flow achieve higher throughput than the other? Try to explain what mechanisms force the two flows to stabilise to the observed throughput.

UDP achieves higher throughput than TCP because it does not use congestion control and continues to send as many packets as it possibly can until it takes up the link bandwidth. TCP on the other hand has congestion control and so limits itself due to the UDP flows' high output. As seen from the graph, the TCP grows relatively quickly from t = 0 to t = 2 due to its slow start mechanisms. However due to UDP's high utilisation of the link, the TCP quickly experiences congestion events and so decreases its congestion window, lowering its throughput. From then the TCP throughput oscillates around 100 packets/s. Although the UDP continues to output as many packets as it can, its throughput oscillates around the 1000 packets/s range due to its unreliability as packets do not always reach the destination.

Question 3: List the advantages and the disadvantages of using UDP instead of TCP for a file transfer, when our connection has to compete with other flows for the same link. What would happen if everybody started using UDP instead of TCP for that same reason?

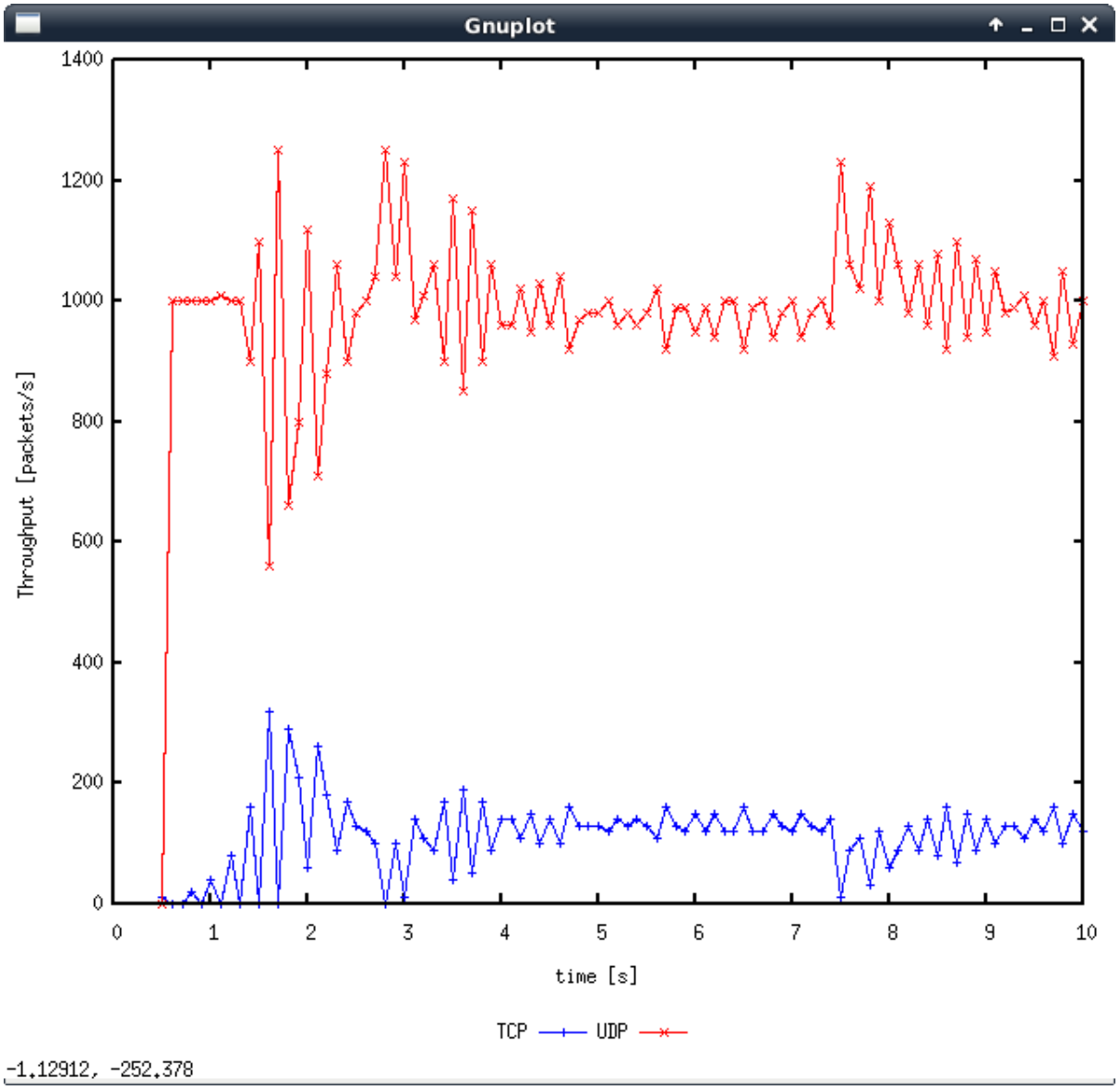
Advantages:

- Less overhead to setup connection (faster)
- Small header size (less data required to transfer files)
- No congestion control allows it to keep sending packets

Disadvantage:

- Cannot determine packet order (so file content may not be arranged in proper order)
- Unreliability (file may be lost in transmission)
- Does not support segmentation (so if file packets are too large for bottleneck link, packet will be dropped)

If everybody started using UDP instead of TCP, there will be large amounts of network traffic as every end system would continue to send as many packets as needed regardless of the network state (due to UDP's lack of congestion control). As such the routers will continue to be overloaded and buffers at routers will continue to overflow causing packet loss and massive delays. This will continue to occur as UDP does not recognise these congestion events and eventually the entire network would be unusable.


Gnuplot
1400
1200
1000
800
600
400
200
0
Throughput [packets/s]
0
1
2
3
4
5
6
7
8
9
10
time [s]
TCP
UDP
-1.12912, -252.378